OULUN SEUDUN
AMMATTIKORKEAKOULU

Mikko Karjalainen

QT-KÄYTTÖLIITTYMÄ VERKKOEMULAATTORILLE

QT-KÄYTTÖLIITTYMÄ VERKKOEMULAATTORILLE

Mikko Karjalainen
Opinnäytetyö
Kevät 2012
Tietotekniikan koulutusohjelma
Oulun seudun ammattikorkeakoulu

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Tietotekniikan koulutusohjelma, ohjelmistokehitys

Tekijä: Mikko Karjalainen
Opinnäytetyön nimi: Qt-käyttöliittymä verkkoemulaattorille
Työn ohjaajat: Pertti Heikkilä (OAMK), Vinski Bräysy (Rugged Tooling Oy)
Työn valmistumislukukausi ja -vuosi: Kevät 2012 Sivumäärä: 30

Opinnäytetyössä toteutettiin käyttöliittymäsovellus Rugged Tooling Oy:n verkkoemulaattorille. Yritys oli kehittänyt sulautetussa laitteessa toimivan tuotteen, johon piti toteuttaa graafinen käyttöliittymä. Työn tilaajana toimi Vinski Bräysy.

Työn tavoitteena oli toteuttaa Windows- ja Linux-käyttöjärjestelmissä toimiva käyttöliittymäsovellus. Sovelluksen avulla piti pystyä luomaan ja lähettämään sääntöjä verkkoemulaattorille. Sääntöjen piti sisältää verkon emulointiin tarvittavat parametrit.

Opinnäytetyössä toteutettu käyttöliittymäsovellus toteutettiin C++-kielellä Qt-ohjelmakirjastoja ja Qt Creator -ympäristöä hyödyntäen. Statistiikan piirto toteutettiin käyttäen Qwt-kirjastoa.

Opinnäytetyön tuloksena kehitettiin vaatimukset täyttävä käyttöliittymäsovellus. Toteutettu käyttöliittymäsovellus testattiin toimivaksi Windows 7- ja Ubuntu Linux -käyttöjärjestelmissä. Toteutetun sovelluksen pohjalta voidaan jatkokehittää erilliset sovellukset sääntötiedostojen luomiseen ja muokkaamiseen ja sääntöjen lähettämiseen.

Asiasanat: Käyttöliittymä, verkkoemulaattori, Qt

ABSTRACT

Oulu University of Applied Sciences
Information Technology, Software Development

Author: Mikko Karjalainen
Title of thesis: Qt User Interface for Network Emulator
Supervisors: Pertti Heikkilä (OUAS), Vinski Bräysy (Rugged Tooling Oy)
Term and year of completion: Spring 2012 Pages: 30

User interface software for network emulator device from Rugged Tooling Ltd was developed in this thesis. Company had developed a product which needed graphical user interface. This thesis was commissioned by Vinski Bräysy.

Purpose of this thesis was to develop user interface software that works in Windows and Linux operating systems. Software needed to be able to create and send rules for network emulator. Rules needed to contain parameters needed in network emulation.

User interface software was developed using C++ language and Qt libraries in Qt Creator environment. Qwt libraries were used in drawing of statistics graphs.

User interface software that fulfills requirements was developed as a result of this thesis. Developed software was tested in Windows 7 and Ubuntu Linux operating systems. Developed software can be used as base of separate applications for creating and modifying and sending rulefiles.

Keywords: User interface, Network emulator, Qt

SISÄLLYS

1 JOHDANTO

Rugged Tooling Oy tekee sulautettuja reaaliaikatyökaluja IP-verkon testaukseen ja monitorointiin. Tämä työ tehtiin yrityksen tarpeeseen saada käyttöliittymä verkkoemulaattorille. Työn tavoitteena oli suunnitella ja toteuttaa verkkoemulaattorin käyttöliittymä C++-kieltä ja Qt-ohjelmakirjastoja hyödyntäen. Käyttöliittymän toteutusvaiheessa käytettiin Git-versionhallintajärjestelmää.

Käyttöliittymän piti toimia Windows- ja Linux-käyttöjärjestelmissä, joten oli luonnollista käyttää Qt-kehitysympäristöä, joka tukee kumpaakin käyttöjärjestelmää. Toteutettu käyttöliittymä toimii yhteytenä käyttäjän ja sulautetun verkkolaitteen välillä. Käyttöliittymän kautta verkkoemulaattorille voidaan syöttää sääntöjä, jotka määrittelevät säännölle tulevalle liikenteelle halutut parametrit, kuten viiveen, viiveen vaihtelun tai pakettien pudotustiheyden.

2 TYÖHÖN LIITTYVIÄ KÄSITTEITÄ

2.1 Verkkoemulaattori

Verkon emulointi on tekniikka, jossa olemassa olevan verkon ominaisuuksia muutetaan hallitusti. Verkkoemulaattori ei emuloi päätelaitteita, vaan verkkoa, johon päätelaitteet kytketään. Verkon emulointi voidaan saavuttaa laittamalla lähiverkkoon laite, joka muuttaa verkon liikennettä emuloidun ympäristön mukaiseksi. Laite voi olla yleiskäyttöinen tietokone, jolla suoritetaan verkon emuloinnin toteuttavaa sovellusta. Laite voi olla myös erityisesti verkon emulointiin tehty. Laite sisällyttää emulaatioon erilaisia verkon parametreja. Näitä parametreja ovat esimerkiksi verkon latenssi, käytettävissä oleva kaista, pakettien tippuminen, pakettien monistaminen, pakettien järjestyksen muuttuminen ja verkon viiveen vaihtelu. Testattavat laitteet liitetään emuloituun ympäristöön, jotta laitteiden ja sovellusten käyttäytymistä voidaan testata oikeankaltaisessa ympäristössä. (1.)

2.2 Qt Creator -kehitysympäristö

Qt Creator on Qt-ohjelmistokehittäjien tarpeisiin suunnattu, monella alustalla toimiva integroitu kehitysympäristö. Qt Creator toimii Windows-, Linux/X11- ja Mac OS X -käyttöjärjestelmissä. Sen avulla ohjelmistokehittäjät voivat toteuttaa sovelluksia useille työpöytä- ja mobiilialustoille. (2.)

2.3 XML-tiedostoformaatti

XML (eXtensible Markup Language) on yksinkertainen ja erittäin joustava tekstiformaatti. XML kehitettiin alun perin vastaamaan suuren mittakaavan elektronisen julkaisun haasteisiin. XML on myös tärkeässä roolissa tiedonvälityksessä verkossa ja muualla. (3.)

XML on alustariippumaton tiedonesitystapa. XML:n avulla voi luoda tietoa, joka voidaan lukea millä tahansa sovelluksella ja alustalla. Tiedon tallentaminen on XML:n itsestäänselvin käyttötarkoitus. (4.)

2.4 Qwt-kirjasto

Qwt-kirjasto (Qt Widgets for Technical Applications) sisältää käyttöliittymäkomponentteja ja apuluokkia pääasiassa teknisten ohjelmien käyttöön. Kaksiulotteisten kuvaajien piirron lisäksi se tarjoaa asteikkoja, liukuja, säätimiä, kompasseja, lämpömittareita, rullia ja nuppeja kontrolloimaan tai esittämään liukulukutyypin arvoja, taulukoita tai asteikkoja. (5.)

2.5 Git-versionhallinta

Git on ilmainen avoimen lähdekoodin hajautettu versionhallinta järjestelmä, joka on suunniteltu käsittelemään kaikkea pienten ja erittäin suurten projektien väliltä nopeasti ja tehokkaasti. Git on suunniteltu toimimaan mahdollisimman tehokkaasti. Git ei ole yksittäinen ohjelma, vaan koostuu suuresta joukosta pienempiä sovelluksia, joista kukin toteuttaa yksittäisiä toimintoja. (6; 7.)

2.6 qmake

qmake on työkalu, joka helpottaa projektin kääntämisprosessia eri alustoilla. qmake automatisoi makefile-tiedoston tekemisen niin, että makefile voidaan tehdä vain muutamalla rivillä tietoa. qmakea voi käyttää vaikka projekti ei olisikaan tehty Qt:lla. qmake tekee makefilen projektitiedoston tietojen perusteella. Projektitiedostot ovat kehittäjän tekemiä ja ne ovat normaalisti melko yksinkertaisia. Monimutkaisemmat projektit voivat vaatia monimutkaisempia projektitiedostoja. (8.)

2.7 Makefile

Makefile on tiedosto, joka kertoo, kuinka systeemi käännetään. Makefile pitää sisällään sääntöjä, muuttujia, käskyjä ja kommentteja. Säännöt ovat ohjeita, mitä tiedostoja käännetään. Säännöt määräävät myös, milloin ja miten tiedostot käännetään. Muuttujaa voidaan käyttää korvaamaan pidempi pätkä tekstiä. Muuttujien käyttäminen tekee makefile-tiedostoista yksinkertaisempia. Käskyjä voidaan käyttää esimerkiksi toisten makefile-tiedostojen mukaan ottamiseen. Käskyjen avulla voidaan myös määritellä käytetäänkö, jotakin osaa makefile-tiedostosta vai ei. Kommentit ovat rivejä, joita ei käsitellä makefileä käytettäessä. (9.)

3 KÄYTTÖLIITTYMÄN VAATIMUKSET

Tässä luvussa esitetään sovellukselle asetetut vaatimukset. Vaatimuksissa määritetään, mitä toiminnallisuuksia toteutettava sovellus sisältää. Vaatimukset on laatinut työn tilannut yritys.

1. Useiden sääntöjen asettaminen: Käyttöliittymässä pitää olla mahdollisuus lisätä useita sääntöjä. Jokaiselle säännölle pitää voida asettaa halutun mukaiset parametrit.

2. Sääntöjen tallentaminen: Käyttöliittymään syötetyt säännöt tulee voida tallentaa uudelleenkäyttöä varten. Säännöt tulee tallentaa helposti käsiteltävään muotoon.

3. Tallennettujen sääntöjen lukeminen: Käyttöliittymään tulee voida lukea aiemmin tallennettuja sääntöjä. Aiemmin tallennettuja sääntöjä pitää pystyä muokkaamaan tarvittaessa.

4. Tallennettujen sääntöjen lähettäminen komentoriviltä: Käyttöliittymällä tehdyt aiemmin tallennetut säännöt tulee voida lähettää suoraan komentoriviltä ilman graafista käyttöliittymää. Käyttöliittymäsovelluksen pitää toimia sekä graafisena että komentorivisovelluksena.

5. Statistiikan näyttäminen: Käyttöliittymän tulee näyttää graafisesti laitteesta saatavaa verkkoliikennestatistiikkaa. Statistiikan piirtäminen tulee voida asettaa päälle ja pois päältä.

6. Usean alustan tukeminen: Käyttöliittymän tulee toimia ainakin Ubuntu Linux- ja Windows 7 -käyttöjärjestelmissä.

7. Tallennettujen sääntöjen tekeminen muilla työkaluilla: Tallennettujen sääntöjen pitää olla sellaisessa muodossa, että niitä voidaan muokata ja tehdä myös muilla työkaluilla.

8. Asetuksien tallentaminen ja lukeminen: Käyttöliittymässä asetettavat asetukset tulee tallentaa sovelluksen sulkeutuessa. Asetukset pitää lukea käynnistyksen yhteydessä, jos asetustiedosto on olemassa. Tallennettavia asetuksia ovat ainakin laitteen IP-osoite ja se onko statistiikan piirto päällä vai pois päältä.

4 KÄYTTÖLIITTYMÄN TOIMINTAYMPÄRISTÖ

Käyttöliittymä kehitettiin Qt Creator -kehitysympäristöllä Windows 7 -käyttöjärjestelmässä. Käyttöliittymän toiminta testattiin myös Ubuntu Linux -käyttöjärjestelmässä.

Käyttöliittymä tehtiin ohjaamaan yrityksen tuotteena olevan verkkoemulaattorin RUDEn (Rugged Deviation Generator) toimintaa. RUDE on sulautettu IP-verkon emulaatioon ja monitorointiin tarkoitettu työkalu. RUDE soveltuu IP-sovellusten ja laitteiden testaukseen. RUDE ei tuota liikennettä, vaan viivästyttää ja muokkaa laitteen läpikulkevaa liikennettä. RUDEssa on varattu kaksi porttia läpikulkevaa liikennettä varten ja neljä toisiinsa kytkettyä porttia liikenteen ja statistiikan monitorointia ja sääntöjen asettamista varten.

Käyttöliittymä toimii Windows- ja Linux-käyttöjärjestelmissä. Käyttöliittymä on yhteydessä verkkoemulaattoriin verkkokaapelilla. Käyttöliittymästä lähetetään verkkoemulaattoriin aloitusviesti ja sääntöviestit. Käyttöliittymästä voidaan lähettää myös verkkoliikennestatistiikkakysely, johon RUDE lähettää vastauspaketin.

Käyttöliittymän ja RUDEn välinen kommunikointi tapahtuu UDP-viestien kautta. Kommunikointia varten toteutettiin seuraavat viestit: sääntöjen muokkauspyyntö, aloituspyyntö, nollauspyyntö, statistiikkapyyntö ja statistiikkavastaus.

5 KÄYTTÖLIITTYMÄN TOTEUTUS

5.1 Dynaamisen kirjaston käyttäminen Windows- ja Linux-käyttöjärjestelmissä

Koska Windows- ja Linux-käyttöjärjestelmissä ei voida käyttää samoja dynaamisia kirjastoja, sovelluksen käännösvaiheessa täytyy ladata dynaamiset kirjastot käyttöjärjestelmäkohtaisesti. Kuvassa 1 on esimerkki siitä, kuinka Qt-projektitiedostossa otetaan käyttöön dynaaminen kirjasto Windows- ja Linux-käyttöjärjestelmissä.

```
unix {
    include( Rude.pri )
    CONFIG+=qwt
}

win32 {
    DEFINES+=QWT_DLL
    INCLUDEPATH+=D:\\qwt-6.0.1\\src
    LIBS+=D:\\qwt-6.0.1\\lib\\libqwtd.a
    CONFIG+=qwt
}
```

KUVA 1. Dynaamisen kirjaston käyttäminen Windows- ja Linux-käyttöjärjestelmissä

Kun Qt-projekti käännetään Linux-ympäristössä, qmake tekee makefilen unix-määritteen sisällä olevan osan mukaan. Windows-ympäristössä qmake käyttää win32-määritteen sisällä olevaa osaa.

5.2 Graafisen sovelluksen ja komentorivisovelluksen yhdistäminen

Käyttöliittymän piti vaatimusten mukaan toimia sekä graafisesti että komentoriviltä ilman graafista käyttöliittymää. Oikea komentorivisovellus tehtäisiin Qt-ympäristössä käyttämällä QCoreApplication-luokkaa, joka ei ole riippuvainen käyttöliittymäkomponentteja sisältävästä QtGui-kirjastosta. Käyttöliittymäsovellus tehdään käyttämällä QApplication-luokkaa. Komentorivisovellus ja graafinen sovellus saatiin samaan sovellukseen siten, että

komentoriviltä argumenttien kanssa ajettaessa sovellus jättää graafisen käyttöliittymän piiloon, lähettää argumentin mukaiset komennot laitteelle ja lopettaa ohjelman suorituksen sen jälkeen. Komentorivisovelluksen luominen esitetään kuvassa 2.

```
QApplication* app;
Rude *rude;
QString default_ip("192.1.1.2");
uint16_t default_port = 7777;

for(int i=1; i<argc; i++)
{
    QString arg(argv[i]);

    if(arg.compare("-x")==0)
    {
        if(i>=argc-3)
        {
            printf("Usage: rude -x filename ip port\n");
            exit(1);
        }
        else
        {
            QString filename = QString(argv[++i]);
            default_ip = QString(argv[++i]);
            default_port = QString("%1").arg(argv[++i]).toUShort();
            app = new QApplication(argc, argv);
            rude = new Rude(1, default_ip, default_port, filename);
            delete rude;
            exit(0);
        }
    }
    else if(arg.compare("-r")==0)
    {
        if(i>=argc-2)
        {
            printf("Sending reset to default (192.1.1.2:7777)\n");
        }
        else
        {
            default_ip = QString(argv[++i]);
            default_port = QString("%1").arg(argv[++i]).toUShort();
            printf("Sending reset (%s:%d)",argv[i-1],default_port);
        }
        app = new QApplication(argc, argv);
        rude = new Rude(0, default_ip, default_port);
        delete rude;
        exit(0);
    }
    else
    {
        printf("Usage:\nrude -r ip port\n");
        printf("rude -x filename ip port\n");
        printf("Gui starts without any arguments.\n");
        exit(1);
    }

}
```

KUVA 2. Komentoriviparametrit ja komentorivisovelluksen luominen

Graafisen käyttöliittymäsovelluksen peruskomponentit Qt-sovelluksessa ovat QApplication ja QMainWindow. Toteutetussa sovelluksessa on Rude-luokka, joka on peritty QMainWindow-luokasta. Rude-luokka on sovelluksen pääluokka,

joka näkyy käyttäjälle sovelluksen pääikkunana. QApplication-luokasta luodaan tarkalleen yksi olio graafisessa käyttöliittymäsovelluksessa. QApplication sisältää tapahtumasilmukan ja suurimman osan sovellukseen liittyvistä asetuksista. Graafisen sovelluksen luominen esitetään kuvassa 3.

```
while(1)
{
    app = new QApplication(argc, argv);
    rude =new Rude(NULL,app);
    rude->setWindowState(rude->windowState() ^ Qt::WindowMaximized);
    rude->show();
    int status = app->exec();
    //If status is not 0, restart application
    if(status==0)
        return status;
    delete rude;
    delete app;
}
```

KUVA 3. Graafisen käyttöliittymäsovelluksen luominen

5.3 Käyttöliittymän yleisrakenne

Käyttöliittymän näkymä luodaan QMainWindow-luokasta perityn Rude-luokan rakentajassa. Rakentajassa asetetaan CentralWidgetiksi QWidget-luokasta luotu myCentralWidget-olio. myCentralWidget-oliolle asetetaan statistiikan piirron ja sääntötaulukon sisältävä asettelu. Käyttöliittymän asetteluun liittyvät koodirivit esitetään kuvassa 4.

```
QVBoxLayout *layout = new QVBoxLayout(this);
layout->addWidget(m_plot);

QVBoxLayout *mainLayout = new QVBoxLayout(this);
mainLayout->addLayout(layout);
mainLayout->addWidget(m_table);

QWidget *myCentralWidget = new QWidget(this);
myCentralWidget->setLayout(mainLayout);

setCentralWidget(myCentralWidget);
```

KUVA 4. Käyttöliittymän asettelu

Käyttöliittymän työkalurivi sisältää painikkeet ja asetuksiin tarvittavat komponentit. Painikkeet luodaan QPushButton-luokasta. Laitteen IP-osoitetta

varten käytetään QLabel- ja QLineEdit-luokkia. Statistiikan piirron asettamiseen käytetään QCheckBox-luokkaa. Työkalurivin luominen on esitetty kuvassa 5. Kuvasta 5 nähdään, että ADD RULE- ja START-painikkeet ovat Rude-luokan jäsenmuuttujia. Kyseiset painikkeet ovat jäsenmuuttujia, koska niiden asetuksia pitää voida helposti muuttaa. RESET-painikeen asetuksia ei tarvitse muuttaa, joten sen ei tarvitse olla Rude-luokan jäsenmuuttuja.

```
addToolBar(m_toolBar = new QToolBar(this));
m_toolBar->setMovable(false);

m_cellLabel = new QLabel(this);
m_cellLabel->setHidden(true);

m_ipAddressLabel = new QLabel("  Device IP address", this);
m_ipAddressLabel->setMinimumSize(100,0);

m_ipAddressEdit = new QLineEdit(m_ipaddress,this);
m_ipAddressEdit->setMaximumSize(135,22);

m_useStatisticsCheckbox = new QCheckBox("Show Statistics", this);
if(m_startStatistics)
{
    m_useStatisticsCheckbox->setChecked(true);
}

m_addRule = new QPushButton("ADD RULE");
m_start = new QPushButton("START");
QPushButton* resetSW = new QPushButton("RESET",this);

m_toolBar->addWidget(m_addRule);
m_toolBar->addWidget(m_start);
m_toolBar->addWidget(resetSW);
m_toolBar->addWidget(m_ipAddressLabel);
m_toolBar->addWidget(m_ipAddressEdit);
m_toolBar->addWidget(m_useStatisticsCheckbox);
```

KUVA 5. Työkalurivin luominen

5.4 Sääntötaulukon luominen

Sääntötaulukko on QTableWidget-luokan olio, johon voidaan lisätä rivi jokaista sääntöä varten. Kuvassa 6 esitetään, kuinka sääntötaulukkoon lisätään oma rivi jokaiselle säännölle. Jokaiselle riville asetetaan aluksi oletusarvot.

```
int i=0;
QTableWidgetItem* item=NULL;
for(i=0;i< m_defaultValues.size();i++)
{
    item = new QTableWidgetItem(m_defaultValues[i]);
    m_table->setItem(m_table->rowCount()-1,i,item);
}
```

KUVA 6. Rivien lisääminen sääntötaulukkoon

Sääntötaulukon jokaiselle sarakkeelle lisätään otsikko luomalla QTableWidgetItem-luokan olio ja asettamalla sille otsikkoon haluttu teksti ja tooltip-teksti. Kuvassa 7 esitetään otsikon lisääminen yhdelle sarakkeelle.

```
QTableWidgetItem *twi = new QTableWidgetItem("Port(s)");
twi->setToolTip("Describes the port that the rule concerns.");
m_table->setHorizontalHeaderItem(i, twi);
m_table->setColumnWidth(i++,55);
```

KUVA 7. QTableWidget-olion lisääminen sääntötaulukkoon otsikoksi

5.5 Sääntöjen tallentaminen ja lukeminen

XML-tiedoston kirjoittamiseen käytetään QtXml-moduulin luokkia QDomDocument ja QDomElement. QDomDocument-luokasta luodaan olio, johon lisätään QDomElement-olioita. Lukeminen tapahtuu vastaavalla tavalla lukemalla QDomElement-olioita QDomDocument-oliosta. Kuvassa 8 lisätään QDomDocument-luokan doc-olioon sääntöihin liittyvät parametrit ja kirjoitetaan ne tiedostoon.

```
QDomElement root;
root = doc.createElement( "items" );
doc.appendChild( root );

int j=0;
for(int i=0; i<m_table->rowCount(); i++)
{
    QDomElement Rule = doc.createElement( "RudeRule" );
    Rule.setAttribute("INPUTport", m_table->item(i,j++)->text());
    Rule.setAttribute("FILTERstring", m_table->item(i,j++)->text());
    Rule.setAttribute("DROPmode", m_table->item(i,j++)->text());
    Rule.setAttribute("DROPratio", m_table->item(i,j++)->text());
    Rule.setAttribute("Bandwidth",m_table->item(i,j++)->text());
    Rule.setAttribute("Bucket",m_table->item(i,j++)->text());
    Rule.setAttribute("DELAYus", m_table->item(i,j++)->text());
    Rule.setAttribute("JITTERus", m_table->item(i,j++)->text());
    Rule.setAttribute("JITTERdist", m_table->item(i,j++)->text());
    Rule.setAttribute("DELAYcorr",m_table->item(i,j++)->text());
    Rule.setAttribute("DELAYskipmode",m_table->item(i,j++)->text());
    Rule.setAttribute("DELAYskipratio",m_table->item(i,j++)->text());
    Rule.setAttribute("SWAPlayers", m_table->item(i,j++)->text());
    Rule.setAttribute("ERRORmode", m_table->item(i,j++)->text());
    Rule.setAttribute("ERRORoffset", m_table->item(i,j++)->text());
    Rule.setAttribute("ERRORvalue", m_table->item(i,j++)->text());
    Rule.setAttribute("ERRORskipmode",m_table->item(i,j++)->text());
    Rule.setAttribute("ERRORskipratio",m_table->item(i,j++)->text());
    Rule.setAttribute("OUTPUTport", m_table->item(i,j++)->text());
    Rule.setAttribute("OUTPUTport2", m_table->item(i,j++)->text());
    Rule.setAttribute("Statistics", m_table->item(i,j++)->text());
    Rule.setAttribute("Rulename", m_table->verticalHeaderItem(i)->text());
    j=0;
    root.appendChild(Rule);
}

QTextStream stream( &file );
QString streamString(doc.toString());
stream << streamString.toUtf8();
file.close();
```

KUVA 8. Sääntöjen parametrien tallennus XML-tiedostoon

5.6 Asetustiedosto

Asetustiedoston luomiseen ja lukemiseen käytetään QSettings-luokkaa. QSettings-luokan avulla saadaan helposti luotua ini-tiedosto, johon voidaan lisätä tarvittava määrä asetuksia. Asetustiedoston lukeminen sovelluksen käynnistysvaiheessa on esitetty kuvassa 9. Sovelluksen käynnistysvaiheessa luetaan laitteen IP-osoite ja statistiikan piirron asetus rude.ini-tiedostosta. Jos tiedostoa ei ole, käytetään oletusasetuksia.

```
QSettings settings("rude.ini", QSettings::IniFormat);
m_ipaddress = settings.value("rude/ipaddress","192.1.1.2").toString();
m_startStatistics = settings.value("rude/use_statistics",1).toInt();
```

KUVA 9. Asetusten lukeminen asetustiedostosta

Sovelluksen sulkemisvaiheessa asetukset talletetaan rude.ini-tiedostoon kuvassa 10 esitetyllä tavalla.

```
QSettings settings("rude.ini",QSettings::IniFormat);
settings.setValue("rude/ipaddress",m_ipAddressEdit->text());
if(m_useStatisticsCheckbox->isChecked())
{
    settings.setValue("rude/use_statistics",1);
}
else
{
    settings.setValue("rude/use_statistics",0);
}
```

KUVA 10. Asetusten tallentaminen asetustiedostoon

5.7 Statistiikan piirtäminen ja Qwt-komponentit

Porttikohtainen statistiikka saadaan verkkoemulaattorilta vastauksena statistiikan kyselysanomaan. Vastaussanomasta parsitaan verkkoliikenteen määrä ja lisätään vastaanotetut arvot kuvaajaan kuvassa 11 esitetyllä tavalla.

```
void Rude::parseStatistics(QByteArray data)
{
    int ruleLength = 32;
    int ruleNumber;
    int ruleOffset = 0;//42;
    int msgHeaderOffset = 16;
    int rules = (data.size()-ruleOffset)/ruleLength;
    const unsigned char *pkt_data;

    MSG_HEADER *msg_header;

    pkt_data = (unsigned char *)data.data();

    msg_header = (MSG_HEADER *)(pkt_data + ruleOffset);
    msg_header->version = qFromBigEndian(msg_header->version);
    msg_header->msg_id = qFromBigEndian(msg_header->msg_id);
    msg_header->port_mask = qFromBigEndian(msg_header->port_mask);
    msg_header->lenght = qFromBigEndian(msg_header->lenght);
    msg_header->kellojakso = qFromBigEndian(msg_header->kellojakso);

    for(ruleNumber = 0; ruleNumber < rules; ruleNumber++)
    {
        STATISTICS *stats;

        stats = (STATISTICS *)(pkt_data + ruleOffset+msgHeaderOffset+ruleLength*ruleNumber);
        stats->rule_number = qFromBigEndian(stats->rule_number);
        stats->flow_count = qFromBigEndian( stats->flow_count);
        stats->packet_count = qFromBigEndian( stats->packet_count);
        stats->error_packet_count = qFromBigEndian( stats->error_packet_count);
        stats->byte_count = qFromBigEndian( stats->byte_count);

        m_plot->addData(m_time, stats->byte_count, stats->rule_number | msg_header->port_mask << 31);
    }
}
```

KUVA 11. Statistiikan vastaussanoman parsimisen toteuttava funktio

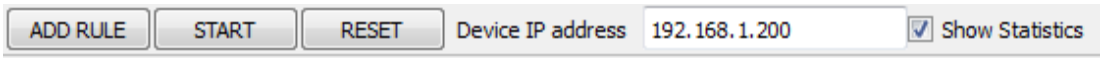
Statistiikan piirtoa varten sovelluksessa on Plot-luokka, joka peritään QwtPlot-luokasta (kuva 12). QwtPlot-luokka sisältää kaksiulotteisen kuvaajan piirtämiseen tarvittavan toiminnallisuuden. Statistiikan piirtoa varten jokaiselle säännölle luodaan oma käyrä QwtPlotCurve-luokasta.

```
class Plot : public QwtPlot
{
    Q_OBJECT
public:
    Plot(QWidget * = 0);
    const QwtPlotCurve *curve(int id) const
        { return data[id].curve; }
    void addData(double data[NRudeData]);
    void addData(long xValue,
                 unsigned long long yValue,
                 unsigned int ruleNumber);
    RuleCurve *addCurve(QString title);

protected:
    void timerEvent(QTimerEvent *e);

private Q_SLOTS:
    void showCurve(QwtPlotItem *, bool on);

private:
    struct
    {
        QwtPlotCurve *curve;
        double data[HISTORY];
    } data[NRudeData];
    double timeData[HISTORY];

    QHash<int, QwtPlotCurve *> curves;

    QHash<int, QVector<long> *> xData;
    QHash<int, QVector<unsigned long long > *> yData;
    QHash<int, QVector<double> *> xDataPerSecond;
    QHash<int, QVector<double> *> yDataPerSecond;

    int dataCount;
};
```

KUVA 12. Plot-luokan määrittely

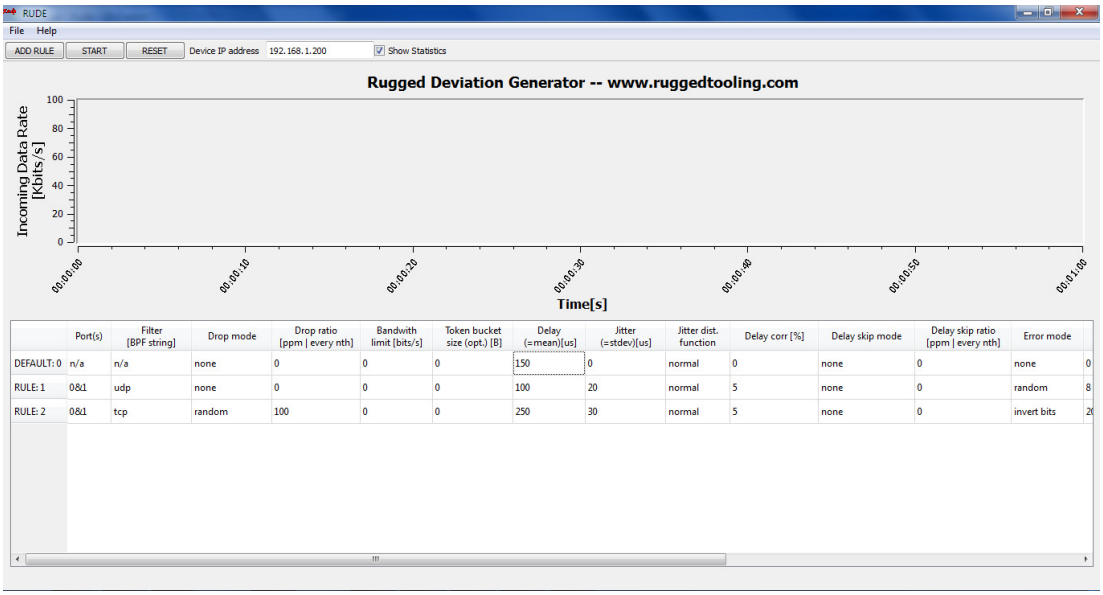
6 KÄYTTÖLIITTYMÄN ESITTELY

Käyttöliittymän työkalurivi sisältää tarvittavat painikkeet, kentän IP-osoitetta varten ja asetuksen statistiikan piirtoa varten. Kuvassa 13 on käyttöliittymän työkalurivi ennen START-painikkeen painamista.



KUVA 13. Työkalurivi ennen START-painikkeen painamista

Kuvasta 14 nähdään, että käyttöliittymä on jaettu kahteen osaan. Ylempänä on statistiikan piirtoon varattu alue ja alempana on sääntötaulukolle varattu alue.

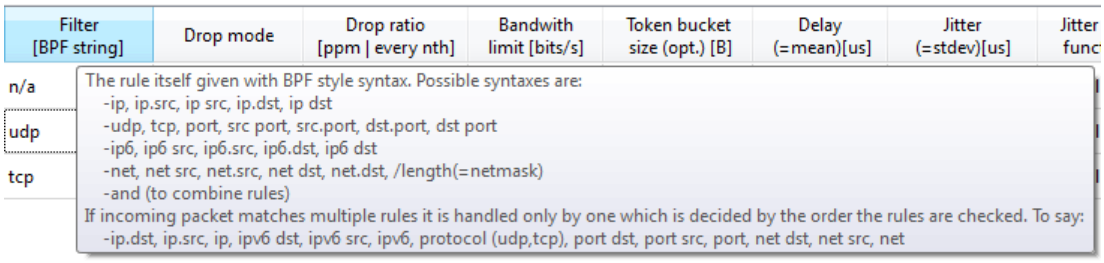


KUVA 14. Yleisnäkymä käyttöliittymästä

6.1 Sääntöjen lisääminen

Sääntötaulukkoon lisätään sääntö ADD RULE -painikkeesta painamalla. Painamisen jälkeen sääntötaulukkoon ilmestyy uusi rivi. Rivillä on sarakkeet kaikille laitteen konfiguroitaville ominaisuuksille. Lisätietoa sarakkeista saadaan näkyviin viemällä osoitin sarakkeen nimen päälle. Kuvassa 15 esitetään

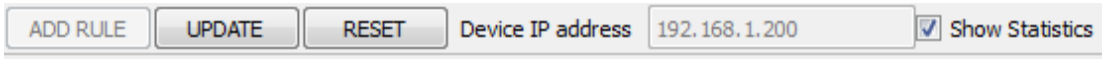
lisätietoa Filter-sarakkeesta. Arvoja muokkaamalla saadaan laite tekemään halutut toiminnot säännön läpi kulkevalle liikenteelle.



KUVA 15. Lisätietoa Filter-sarakkeesta

6.2 Sääntöjen muuttaminen

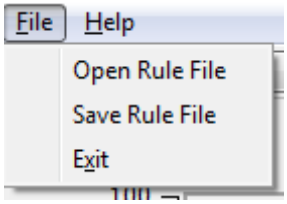
Sääntöjä ei voi lisätä tai poistaa sääntöjen lähettämisen jälkeen ilman laitteen ja käyttöliittymän nollausta. Kuvasta 16 nähdään kuinka ADD RULE -painike muuttuu ei aktiiviseksi. Nollaus tapahtuu painamalla RESET-painiketta. Painikkeen painamisen jälkeen käyttöliittymäsovellus lähettää laitteelle nollaussanoman ja käynnistyy uudelleen. Sääntöihin liittyviä parametreja voidaan muuttaa sääntöjen aloitussanoman lähettämisen jälkeen muokkaamalla muutettavan parametrin arvo halutuksi ja painamalla UPDATE-painiketta.



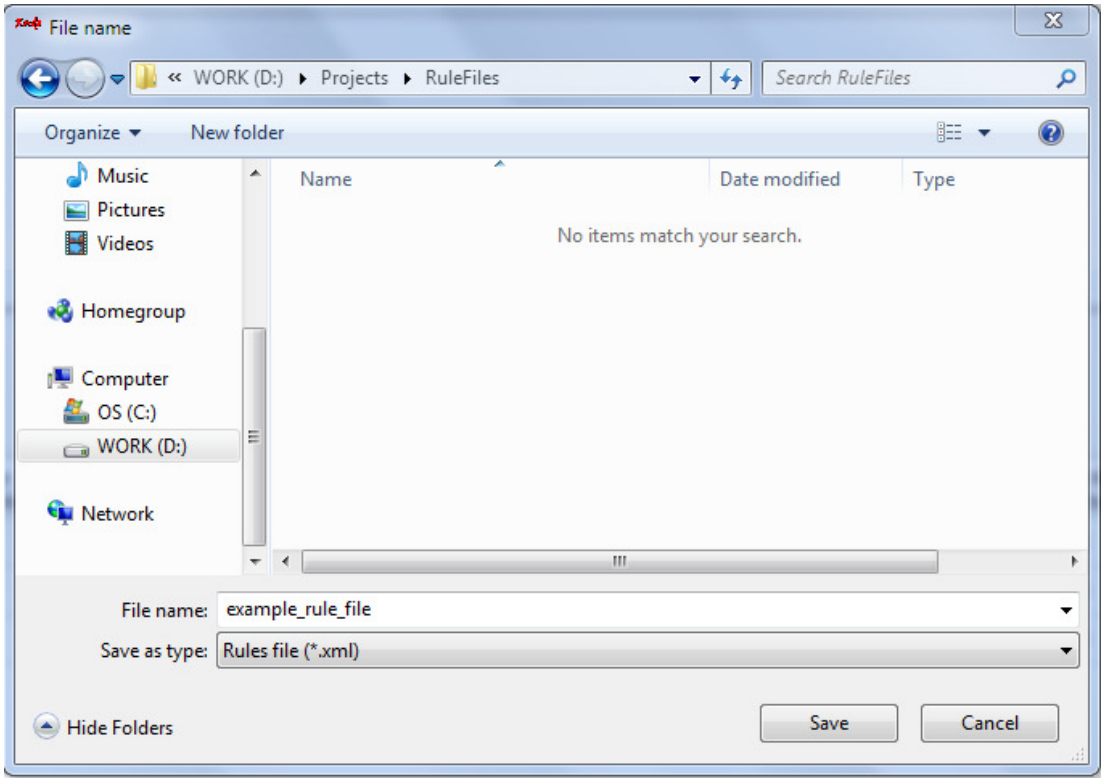
KUVA 16. Työkalurivi START-painikkeen painamisen jälkeen

6.3 Sääntöjen tallentaminen ja uudelleenkäyttäminen

Lisätyt säännöt voidaan tallentaa valitsemalla File-valikosta Save Rule File. Valinnan jälkeen avautuu dialogi, jossa annetaan tallennettavan sääntötiedoston nimi ja sijainti. Kuvassa 17 esitetään File-valikon sisältö ja kuvassa 18 tiedostodialogi.



KUVA 17. File-valikko



KUVA 18. Tiedostodialogi

Tämän jälkeen käyttöliittymäsovellus tallentaa säännöt annetun nimiseen XML-tiedostoon. Tiedoston rakenne on esitetty kuvassa 19. Aiemmin tallennetut säännöt voidaan ladata uudelleenkäytettäväksi valitsemalla File-valikosta Load Rule File. Tiedoston valinnan jälkeen käyttöliittymäsovellus lukee tiedostosta aiemmin talletetut säännöt käyttöliittymän sääntötaulukkoon. Kuvassa 20 on esitetty kuvan 19 säännöt käyttöliittymässä.

```
<!DOCTYPE RudeRulesData>
<items>
  <RudeRule Bandwidth="0" Bucket="0" JITTERdist="normal" Rulename="DEFAULT: 0" DROPratio="0" ERRO
  <RudeRule Bandwidth="0" Bucket="0" JITTERdist="normal" Rulename="RULE: 1" DROPratio="0" ERRORsk
  <RudeRule Bandwidth="0" Bucket="0" JITTERdist="normal" Rulename="RULE: 2" DROPratio="100" ERROR
</items>
```

KUVA 19. Esimerkki XML-tiedoston rakenteesta

	Port(s)	Filter [BPF string]	Drop mode	Drop ratio [ppm | every nth]	Bandwith limit [bits/s]	Token bucket size (opt.) [B]	Delay (=mean)[us]	Jitter (=stdev)[us]	Jitter dist. function	Delay corr [%]	Delay skip mode	Delay skip ratio [ppm | every nth]	Error mode
DEFAULT: 0	n/a	n/a	none	0	0	0	150	0	normal	0	none	0	none
RULE: 1	0&1	udp	none	0	0	0	100	20	normal	5	none	0	random
RULE: 2	0&1	tcp	random	100	0	0	250	30	normal	5	none	0	invert bits

KUVA 20. Esimerkki säännöistä käyttöliittymässä

6.4 Sääntöjen lähettäminen

Säännöt lähetetään laitteelle painamalla START-painiketta. Painamisen jälkeen käyttöliittymäsovellus tarkistaa, että sääntöjen syntaksi on oikea. Sääntöjä ei lähetetä, jos jossain säännössä on virhe. Virheellisten sääntöjen Filter-sarake muuttuu punaiseksi ja virheettömien sääntöjen vihreäksi (kuva 21).

	Port(s)	Filter [BPF string]
DEFAULT: 0	n/a	n/a
RULE: 1	0&1	udp
RULE: 2	0&1	sctp

KUVA 21. Esimerkki virheellisestä säännöstä

Jos kaikki säännöt ovat virheettömiä, käyttöliittymä lähettää laitteelle aloitussanoman ja sääntöjen muokkaussanomat. Tiedostoon talletetut säännöt voidaan lähettää laitteelle myös ajamalla käyttöliittymäsovellus komentoriviltä kuvassa 22 esitetyllä komennolla.

```
Rude -x [rulefile] [ip] [port]
```

KUVA 22. Talletettujen sääntöjen lähettäminen komentoriviltä

6.5 Laitteen IP-osoitteen asettaminen

Käyttöliittymässä voidaan asettaa IP-osoite, johon säännöt ja komennot lähetetään. Asetettu IP-osoite talletetaan asetustiedostoon, kun käyttöliittymäsovellus suljetaan. Seuraavalla käynnistyskerralla talletettu IP-osoite luetaan asetustiedostosta.

6.6 Statistiikan näyttäminen

Laitteelta saatavasta statistiikasta piirretään kaksiulotteinen graafi käyttöliittymän yläosassa. Graafin x-akselilla on aika sekunteina ja y-akselilla datanopeus kilobitteinä sekunnissa. Statistiikan piirtäminen voidaan asettaa päälle valitsemalla käyttöliittymän työkaluriviltä Show Statistics. Statistiikan piirtämisen asetus tallennetaan asetustiedostoon sulkemisvaiheessa ja luetaan sovelluksen käynnistyksen yhteydessä.

6.7 Asetusten tallentaminen

Käyttöliittymäsovellusta suljettaessa sovellus tallettaa sen hetkisen laitteen IP-osoitteen ja tiedon siitä, onko statistiikkakysely päällä, rude.ini-asetustiedostoon. Kuvassa 23 esitetään asetustiedoston sisältö.

```
[rude]
ipaddress=192.168.1.200
#1 = statistics on, 0 = statistics off
use_statistics=1
[/rude]
```

KUVA 23. Asetustiedoston sisältö

7 YHTEENVETO

Tämän opinnäytetyön tarkoituksena oli toteuttaa käyttöliittymä yrityksen tuotteena olevalle verkkoemulaattorille. Käyttöliittymän piti toimia Windows- ja Linux-käyttöjärjestelmissä. Työn tuloksena syntyi tilaajan vaatimukset täyttävä käyttöliittymäsovellus.

Käyttöliittymän jatkokehitykseen on tullut käytön myötä erilaisia vaihtoehtoisia ideoita. Ensimmäinen vaihtoehto on tehdä sääntöjen lähetykseen oma sovellus, jolloin käyttöliittymän tarkoitukseksi jäisi ainoastaan lähetettävien sääntö-tiedostojen teko ja mahdollisesti verkkoliikennestatistiikan esittäminen graafisesti. Verkkoliikennestatistiikkaa voitaisiin lähettää myös esimerkiksi Netflow-formaatissa, jolloin sen esittämiseen voitaisiin käyttää myös muita sovelluksia.

Toinen vaihtoehto on laittaa verkkoemulaattoriin Linux-käyttöjärjestelmä ja WWW-serveri ja tehdä web-pohjainen käyttöliittymä, jolloin verkko-emulaattorista saadaan itsenäinen verkon yli käytettävä laite. Tämän vaihtoehdon huonona puolena on, että käyttöjärjestelmän ja WWW-serverin tarvitsemat resurssit ovat pois laitteen muusta suorituskyvystä. Jos siirrytään käyttämään tehokkaampaa verkkolaitetta, tällä ei ole niin suurta merkitystä.

LÄHTEET

1. Network emulation. 2012. Saatavissa: http://en.wikipedia.org/wiki/Network_emulation. Hakupäivä 21.5.2012.

2. Qt Creator IDE and Tools. 2012. Saatavissa: http://qt.nokia.com/products/developer-tools. Hakupäivä 21.5.2012.

3. Extensible Markup Language (XML). 2012. Saatavissa: http://www.w3.org/XML/. Hakupäivä 21.5.2012.

4. Net to XML. 2012. Saatavissa: http://www.ibm.com/developerworks/xml/newto/. Hakupäivä 21.5.2012.

5. Qwt - Qt Widgets for Technical Applications. 2012. Saatavissa: http://qwt.sourceforge.net/. Hakupäivä 21.5.2012.

6. git. 2012. Saatavissa: http://git-scm.com/. Hakupäivä 21.5.2012.

7. Git. 2012. Saatavissa: http://fi.wikipedia.org/wiki/Git. Hakupäivä 21.5.2012.

8. qmake manual. 2012. Saatavissa: http://doc.qt.nokia.com/4.7-snapshot/qmake-manual.html. Hakupäivä 21.5.2012.

9. GNU make - 3 Writing Makefiles. 2012. Saatavissa: http://www.gnu.org/software/make/manual/make.html#Makefiles. Hakupäivä 24.5.2012.